|  |  |
| --- | --- |
|  |  |

EPTF CLL Scheduler, Function Description

Contents

[1 Introduction 2](#_Toc437000140)

[1.1 Revision history 2](#_Toc437000141)

[1.2 How to read this document 2](#_Toc437000142)

[1.3 Scope 2](#_Toc437000143)

[1.4 Recommended way of reading 2](#_Toc437000144)

[1.5 Typographical conventions 2](#_Toc437000145)

[1.6 Abbreviations 2](#_Toc437000146)

[1.7 Terminology 3](#_Toc437000147)

[2 General description 3](#_Toc437000148)

[3 Functional interface 4](#_Toc437000149)

[3.1 Naming conventions 4](#_Toc437000150)

[3.2 Public functions 4](#_Toc437000151)

[3.2.1 Initialization 4](#_Toc437000152)

[3.2.2 Scheduling an action 4](#_Toc437000153)

[3.2.3 Cancelling an action 5](#_Toc437000154)

[3.2.4 Action handler 5](#_Toc437000155)

[3.2.5 Load measurement 5](#_Toc437000156)

[3.3 Miscellaneous functions 6](#_Toc437000157)

[3.4 Summary table of all public functions for EPTF Scheduler 7](#_Toc437000158)

[4 DTE handling 7](#_Toc437000159)

[5 References 7](#_Toc437000160)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2007-12-01 | PA1 | First draft version | EGBOTAT |
| 2007-12-06 | PA2 | Final version after review | EGBOTAT |
| 2011-09-29 | PB1 | FBQ removed | ETHJGI |
| 2014-10-08 | PC1 | Added section DTE handling and updated section Scheduling an action | EDORNAG |
| 2014-11-21 | PC2 | Added comment on DTE handling | ESZILSZ |
| 2014-12-02 | C | Updated for release | ESZILSZ |
| 2015-08-12 | PD1 | Load measurement added | EZOLZSI |
| 2015-08-17 | PD2 | References chapter updated | ESZILSZ |
| 2015-12-03 | D | Updated for release | ESZILSZ |

## How to read this document

This is the Function Description for the Scheduler of the Ericsson Performance Test Framework (EPTF), Core Library (CLL). EPTF CLL is developed for the TTCN-3 [1] Toolset with TITAN [2]. For more information on the EPTF CLL, please consult the Product Revision Information [3].

## Scope

This document is to specify the content and functionality of the Scheduler feature of the EPTF CLL.

## Recommended way of reading

The readers are supposed to get familiar with the concept and functionalities of EPTF CLL [4]. They should get familiar with the list of acronyms and the glossary in Section 1.6 and 1.7, respectively.

## Typographical conventions

Important concepts are denoted by *italic* font wherever they are first used in the given context.

## Abbreviations

CLL Core Library

EPTF Ericsson Load Test Framework, formerly TITAN Load Test Framework

TitanSim Ericsson Load Test Framework, formerly TITAN Load Test Framework

TTCN-3 Testing and Test Control Notation version 3 [1]

## Terminology

*EPTF Core Library (CLL)* is that part of the TitanSim software that is totally project independent. (That is, which is not protocol-, or application-dependent). The EPTF CLL is to be supplied and supported by the TCC organization. Any EPTF CLL development is to be funded centrally by Ericsson

*Red Black Tree* The EPTF Red Black Tree feature is a self-balancing binary search tree. It can be used to store indexes into an associated array.

# General description

This document specifies the Scheduler feature of the EPTF CLL.

The EPTF Scheduler feature makes it possible to schedule events (actions) that should happen at a given time measured from the start of the component or relative to the schedule of the event. The Scheduler uses only two TTCN-3 timers to implement scheduling. These are the component clock – which is started at initialization – and the next event timer.

To be able to use EPTF Scheduler, the user component should extend the EPTF\_Scheduler\_CT component.

The EPTF Scheduler uses the EPTF Red Black Tree (see [6]) to manage its database of events. To use the Scheduler, import the following modules:

* EPTF\_CLL\_Scheduler\_Definitions
* EPTF\_CLL\_RBTScheduler\_Definitions
* EPTF\_CLL\_RBTScheduler\_Functions

The main altstep of the Scheduler is activated as default on initialization, and performs all necessary steps of handling actions automatically. A typical workflow using the functionality of the Scheduler is the following:

* Initialization of the Scheduler
* scheduling events, and specifying their handler callback function reference
* handling events in their respective handler functions; handled events are automatically removed from the event queue
* cancelling pending events by their event index if needed (for example during cleanup)

# Functional interface

Apart from this description a cross-linked reference guide for the EPTF CLL Functions can be reached for on-line reading [5].

## Naming conventions

All functions have the prefix f\_EPTF\_Scheduler\_.

Obsolete functions have the prefix f\_EPTF\_SchedulerComp\_, for example f\_EPTF\_SchedulerComp\_refreshEventTimer.

## Public functions

### Initialization

Before using the EPTF Scheduler functions the

*f\_EPTF\_Scheduler\_init\_CT ()*

function should be called. This initializes the EPTF Scheduler feature.

A function with an obsolete name for the same functionality:

*f\_EPTF\_SchedulerComp\_InitScheduler()*

### Scheduling an action

*f\_EPTF\_SchedulerComp\_scheduleAction(pl\_when, pl\_actionHandler, pl\_action, pl\_eventIndex, pl\_roundIt, pl\_dteHandler)*

This function can be used to register a new event to the event queue. The event is executed at the time when the component time reaches the value given in pl\_when. The ‘pl\_when’ argument is rounded if pl\_roundIt is true. At the execution time the hander given in pl\_actionHandler is called with the parameters given in pl\_action. The index of the event is returned in the parameter pl\_eventIndex, and can be used, for example, to cancel the event later. Returns true on success, false on error.

The argument pl\_dteHandler defines the handler function that will be called when DTE happens during the execution of the action. Default: null, that is, no handler is called.

### Cancelling an action

*f\_EPTF\_SchedulerComp\_CancelEvent(qidx)*

This function can be used to remove the event from the event queue. The function fails and returns false if the event at the specified index is not valid.

### Action handler

The following private altstep is the main event handler of the Scheduler:

*as\_EPTF\_SchedulerComp\_ActionHandler()*

The initialization function activates it as default, placing it in the main alt-loop of the component that extends the Scheduler.

*f\_EPTF\_SchedulerComp\_performActions()*

This private function handles all actions that has a ‘when’ parameter lower than or equal to the snapshot time taken from the component clock. It is called automatically by as\_EPTF\_SchedulerComp\_ActionHandler

### Load measurement

*f\_EPTF\_Scheduler\_enableLoadMeasurement(pl\_enable)*

This function can be used to enable/disable load measurement for overload detection. The load is measured as the cumulative sum of lag of events during the load measurement period.

*f\_EPTF\_Scheduler\_setLoadMeasurementPeriod(pl\_loadMeasurementPeriod)*

This function can be used to set the load measurement time period (seconds).

*f\_EPTF\_Scheduler\_setMaxLoadThreshold(pl\_maxLoadThreshold)*

This function can be used to set the maximal load threshold When the load exceeds this value, the Scheduler actively controls the overload situation by scheduling events in the future only. If maxLoadThreshold is greater than 1.0, there is no overload control.

*f\_EPTF\_Scheduler\_getLoad()*

This function can be used to get the last measured load value. Its return value is between 0.0 (no load) and 1.0 (full load).

*f\_EPTF\_Scheduler\_getLoadMeasurementInterval()*

This function can be used to get the last load measurement interval. If load measurement is not enabled, its return value is -1.0.

## Miscellaneous functions

*f\_EPTF\_SchedulerComp\_refreshEventTimer()*

This private function sets the event timer to the time of the next scheduled action. It restarts the timer if necessary. This function is automatically called in as\_EPTF\_SchedulerComp\_ActionHandler, f\_EPTF\_SchedulerComp\_scheduleAction and f\_EPTF\_SchedulerComp\_CancelEvent.

*f\_EPTF\_SchedulerComp\_refreshSnapshotTime()*

This function reads the component clock and stores the time in v\_EPTF\_snapshotTime component variable. Proposed usage is to call it in the beginning of any altstep where v\_EPTF\_snapshotTime is used for example to schedule a new event.

*f\_EPTF\_SchedulerComp\_refreshSnapshotTime()*

Returns the snapshot time, that is, the value of the component clock at the time when *f\_EPTF\_SchedulerComp\_refreshSnapshotTime()* was called.

*f\_EPTF\_SchedlerComp\_schedulerIsNotLate()*

This private function returns true if the head event within the event queue is scheduled later than the actual value of the component clock, otherwise returns false.

*f\_EPTF\_SchedulerComp\_eventQueueIsNotEmpty()*

A private function to check whether the event queue is not empty. Returns true if the event queue is not empty, false otherwise.

*f\_EPTF\_SchedulerComp\_eventQueueIsEmpty()*

A private function to check whether the event queue is empty. Returns true if the event queue is empty, false otherwise.

*f\_EPTF\_SchedulerComp\_eventIsValid(pl\_qidx)*

Function to check whether an event is valid within the event queue. Returns true if the event is valid. An event is invalid during it is executed. After executing the event, the event is automatically cancelled (that is, deleted).

*f\_EPTF\_SchedulerComp\_eventIsInvalid(pl\_qidx)*

Function to check whether an event is invalid within the event queue. Returns true if the event is invalid.

## Summary table of all public functions for EPTF Scheduler

Table 1 - Summary of Scheduler functions

|  |  |
| --- | --- |
| Function name | Description |
| f\_EPTF\_Scheduler\_init\_CT | initialization |
| f\_EPTF\_SchedulerComp\_scheduleAction | schedule an event |
| f\_EPTF\_SchedulerComp\_CancelEvent | cancel an existing event |
| f\_EPTF\_SchedulerComp\_refreshSnapshotTime | refresh the snapshot time |
| f\_EPTF\_SchedulerComp\_snapshotTime | returns the snapshot time |
| f\_EPTF\_SchedulerComp\_eventIsValid | check if an event is valid |
| f\_EPTF\_SchedulerComp\_eventIsInvalid | check if an event is invalid |
| f\_EPTF\_Scheduler\_enableLoadMeasurement | enables/disables load measurement |
| f\_EPTF\_Scheduler\_setLoadMeasurementPeriod | set the load measurement period |
| f\_EPTF\_Scheduler\_setMaxLoadThreshold | set the maximal load value |
| *f\_EPTF\_Scheduler\_getLoad* | get the last measured load value. |
| *f\_EPTF\_Scheduler\_getLoadMeasurementInterval* | get the last load measurement interval |

# DTE handling

When the scheduler executes the action handler at the scheduled time and a dynamic test case error occurs, the scheduler does not stop the execution of the component but prints out a warning message about the error. It is also possible to specify an error handler function in f\_EPTF\_SchedulerComp\_scheduleAction. That handler is called when the DTE occurs and receives the event parameters and the error string. When no handler is specified (which is the default) only the warning message is displayed about the error and the scheduler goes on to execute the next event.

By default DTE handling is disabled.

# References

1. ETSI ES 201 873-1 v3.2.1 (2007-02)  
   The Testing and Test Control Notation version 3. [Part 1: Core Language](http://www.etsi.org/deliver/etsi_es/201800_201899/20187301/03.02.01_60/es_20187301v030201p.pdf)
2. 1/198 17-CRL 113 200/5 Uen  
   User Guide for TITAN TTCN-3 Test Executor
3. 109 21-CNL 113 512-22 Uen   
   EPTF CLL for TTCN-3 toolset with TITAN, Product Revision Information
4. 155 17-CNL 113 512 Uen   
   EPTF CLL for TTCN-3 toolset with TITAN, Function Specification
5. EPTF CLL for TTCN-3 toolset with TITAN, [Reference Guide](http://ttcn.ericsson.se/TCC_Releases/Libraries/EPTF_Core_Library_CNL113512/doc/apidoc/html/index.html)
6. 12/155 16-CNL 113 512 Uen  
   EPTF CLL Red Black Tree, Function Description